(12)

(21) **2 468 351**

(51) Int. Cl.7 : **H04L 9/32**

(22) **16.06.2004**

(71)
SXIP NETWORKS INC.,
206 - 55 Water Street, VANCOUVER, B1 (CA).

HARDT, DICK C. (CA).
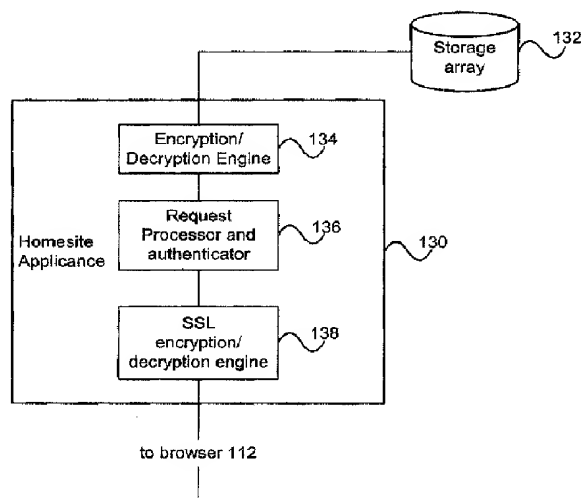
(74)
BORDEN LADNER GERVAIS LLP

(72)

(54) MECANISMES D'AUTHENTIFICATION DE SYSTEME DE GESTION D'IDENTITE HIERARCHIQUE REPARTIE
(54) DISTRIBUTED HIERARCHICAL IDENTITY MANAGEMENT SYSTEM AUTHENTICATION MECHANISMS

(57)
    A set of methods, and systems, for use in an identity management system are disclosed herein.

(54) Titre : MECANISMES D'AUTHENTIFICATION DE SYSTEME DE GESTION D'IDENTITE HIERARCHIQUE REPARTIE
(54) Title: DISTRIBUTED HIERARCHICAL IDENTITY MANAGEMENT SYSTEM AUTHENTICATION MECHANISMS
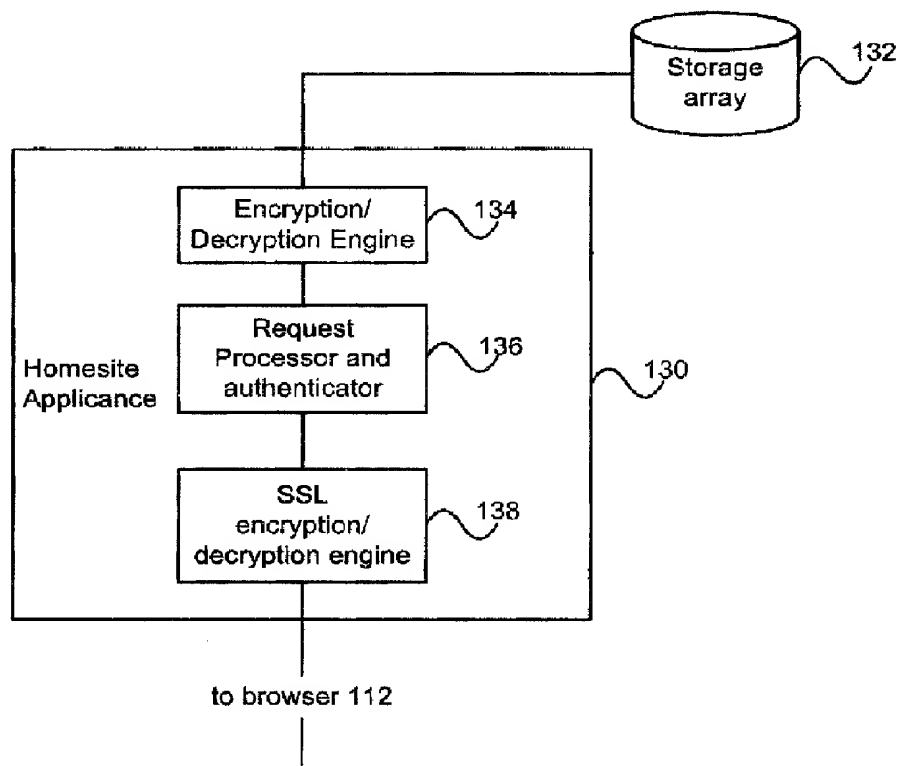
(57) Abrégé/Abstract:
A set of methods, and systems, for use in an identity management system are disclosed herein.

ABSTRACT

A set of methods, and systems, for use in an identity management system are disclosed herein.

DISTRIBUTED HIERARCHICAL IDENTITY MANAGEMENT SYSTEM AUTHENTICATION MECHANISMS

FIELD OF THE INVENTION

The present invention relates generally to electronic identity management systems. More particularly, the present invention relates to authentication and security for data exchange in a distributed hierarchical identity management system pertaining to auditable privacy policies.

BACKGROUND OF THE INVENTION

Many identity management systems are known in the art including Microsoft's Passport™, the Liberty Alliance, and the identity management system described in detail in Canadian Patent No. 2,431,311 and its corresponding applications including U.S. Patent Application Publication No. US 2003/0229783A1, each owned by Sxip Networks.

Identity management is of great concern in the electronic commerce field because it allows an electronic vendor or service provider to provide a user with customized services and further allows the vendor to track user behavior. From the user perspective, identity management permits a user to retrieve previously stored identifying information. In the absence of any identity management system each individual use of a service would be accompanied by selecting a series of settings that could not be maintained in the following session. For example, in the case of an on-line purchase the lack of an identity management system would entail requiring a user to provide all identifying information, including credit card numbers, billing and shipping addresses for each and every purchase. This is commonly regarded as a nuisance to consumers, and serves as a detriment to on-line commerce.

One skilled in the art will appreciate that though the following discussion generally relates to electronic commerce applications of identity management, a number of other non commerce related services, such as customisation of a news site, registration for access to a newspaper, and maintenance of a desired display configuration for an electronic discussion site, all make use of identity management. For these services the ability of a user to have a single-sign on for multiple services is a high priority.

To ensure that a user can use a single user ID for a plurality of services, and to provide what is commonly referred to in the art as "single sign-on", a second generation identity management and authentication solution was created. This solution is typically referred to as a hierarchical identity management system, and is typified by Microsoft's Passport™ service.

In contrast to the hierarchical nature of the Passport™ service, Liberty Alliance systems are referred to as distributed. These distributed systems have a plurality of identity storage providers, and vendor sites communicate to the identity storage provider of their user through a complex secured communication back channel that employs a number of different cryptographic techniques requiring the maintenance of complex and large key rings.

However, according to the specification of the Liberty Alliance, only a defined set of user identity information is stored at the identity storage providers. Thus, E-tailers must still question users about information not stored by an identity storage provider. Additionally, public and private encryption keys require each identity storage provider to be able to perform numerous computation intensive tasks for each data request. Additionally, a sophisticated key management system must be employed as the size of the web of trust increases. One skilled in the art will readily appreciate that the number of identity storage providers in a web of trust cannot scale infinitely. Though it is possible to implement a system whereby each identity storage provider can trust every other identity storage provider when there is a small number of identity storage providers, it is unlikely that such a system can be implemented in a reliable fashion when the number of identity storage providers scales into the tens of thousands.

From the perspective of user, the model presented by the Liberty Alliance has a number of drawbacks. A user's single sign-on abilities are somewhat restricted. A user is assigned a unique user ID that identifies them to their selected identity storage provider.

Both Passport™ and the Liberty Alliance provide E-tailers and other sites requiring user authentication with pair-wise unique identifiers (PUIDs). PUIDs allow the e-tailer to store information and build a profile on a user, while preventing two e-tailers from easily correlating their databases to determine user activities and patterns. PUIDs in the liberty alliance are assigned by the identity storage provider holding the user profile, and cannot be matched to the user account by any other identity storage provider, thus if a user

- 2 -

chooses to change identity storage providers all the site specific settings at each E-tailer are lost. This handcuffs users to an identity storage provider providing no more opportunity for portability for most users than the single source Passport™ does. Furthermore, the purpose of the PUID assigned by either Passport™ or the Liberty Alliance can be overcome by correlating other information such as credit card information.

As described above, conventional identity management systems employ back channels between the identity data providers and other sites in the networks. These back channels are provided as a one size fits all solution. Thus, in order to provide security for sensitive information, the back channel is encrypted. If the identity management system is distributed, the channels make use of complex and unwieldy key rings to ensure security. However, if the only data that the user is submitting is non-sensitive, such as e-mail or postal address information, the data does not require a secure channel. Many sites requesting this information do not operate on secure servers and moving to such a system would be considered onerous. It would be desirable to offer a mechanism to allow a user to interact with the data store in a secure fashion to ensure protection of authentication information, but to then submit information to other websites over insecure channels, through preferably only after user approval.

In addition to requiring a secure server, conventional identity management systems requiring the scripts or code operating at a vendor website to be modified to interact with the defined channels and data passing techniques specified by the network. Though this may not be a problem for new, well funded entities, changing a large established code base is a daunting task if the site is already live and functioning. Thus, it would be beneficial to provide a system to allow an online merchant to very slightly modify an existing site to join an identity management network.

Conventional identity management networks provide no mechanisms to easily incorporate a website's existing user base. Conventional identity management systems require users to lose their history at a website to use the identity management network in conjunction with the website at which an existing account is established. This is a detriment to a website operator interested in tracking the historical behaviour of users. It would be advantageous to offer a mechanism to allow user history and preferences to be maintained for existing users of a website as they join the identity management network.

- 3 -

In conventional hierarchical identity management systems, all user data is centrally stored, thus creating a central location from which data can be obtained if the system is compromised. Furthermore, users must have a level of trust with the central party so that they trust that the stored data will not be leaked or otherwise revealed. In a distributed system, there is a plurality of user data stores. If a single store is compromised only the data about its users is revealed.

In both cases the user data can be encrypted prior to storage to allow for added security in the event that the data store is compromised. However, the user must still have a trust relationship with the data store operator to know that the data will not be decrypted inappropriately. Further, the user must trust the data store operator is capable of securely storing the decryption key so that it is not made available inadvertently.

Furthermore, in a distributed identity management system the ability to increase the number of identity service providers is important, so that users have a variety of parties to choose from, preferably some of those entities have a pre-existing trust relationship with the user. However, as with all complex systems and networks, it is difficult and time consuming to create a secured server to store user data and conform to the requirements of the identity management network. Thus, it would be desirable to provide an identity management server that requires a minimal amount of user customization, and allows the secure storage of user information so that the data cannot be inadvertently, or otherwise, released by the identity management system without user approval.

As identity management systems begin use in a variety of fields and niche markets, new security and authentication mechanisms, along with other such advances, become necessary to account for previously unforeseen needs. It is, therefore, desirable to provide an identity management system with novel authentication and security mechanisms.

SUMMARY OF THE INVENTION

It is an object of the present invention to obviate or mitigate at least one disadvantage of previous identity management systems.

In a first aspect of the present invention, there is provided a homesite appliance for managing user identity information. The homesite appliance comprises a request

- 4 -

processor. The request processor receives requests for the user identity information from an external system, authenticates a user associated with the user identity information and obtains authorization from the authenticated user to release the requested user identity information, extracts the requested user identity information from a schema containing the user identity information, and transmits the extracted user identity information to the external system upon receipt of user authorization.

In an embodiment of the present invention, the homesite appliance further includes a storage array interface for connecting the request processor to a storage array, the storage array for storing the schema containing the user identity information sent by the request processor, and for retrieving previously stored user identity information, preferably the storage array is external to the homesite appliance, and the appliance includes a hardware encryption engine for encrypting user identity information sent to the storage array. In another embodiment, the hardware encryption engine includes means for decrypting user identity information retrieved from the storage array, and the homesite appliance further includes a cache memory for use by the request processor to cache decrypted user identity information, and a cache memory controller for directing retrieval requests from the request processor to one of the cache memory and the external storage array. In another embodiment, the request processor includes user authentication means for authenticating the identity of the user using at least one of a request for a username and password combination, a request for an out of band authentication request for a shared secret, a request for biometric data and a request for a securely generated token. In a further embodiment, there is included means to receive requests for the user identity information from the external system via the user associated with the user identity information. In another embodiment, the request processor includes means to transmit the retrieved user identity information to the external system via the user associated with the user identity information. In a further embodiment the request processor includes means to encrypt the retrieved user identity information prior to transmission to the user, wherein the means to encrypt preferably perform secure sockets layer encryption or wherein the appliance includes a secure sockets layer encryption engine for encrypting the retrieved user identity information prior to transmission. In a further embodiment, the hardware encryption engine includes means to encrypt the retrieved user identity information prior to transmission to the user, wherein the means to encrypt the retrieved

- 5 -

user identity information preferably perform secure sockets layer encryption. In another embodiment, the homesite appliance includes an authoritative site interface, for obtaining and refreshing third party assertions associated with the user identity information.

Other aspects and features of the present invention will become apparent to those ordinarily skilled in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the present invention will now be described, by way of example only, with reference to the attached Figures, wherein:

Figure 1 is a block diagram illustrating dataflow in an embodiment of the identity management system of the present invention;

Figure 2 is a block diagram illustrating the datapaths used to enable a secure connection between the browser and the homesite for an insecure connection to a membersite;

Figure 3 is a block diagram illustrating the dataflow and network configuration for using a simple site server;

Figure 4 is a block diagram illustrating the dataflow and network configuration for using a browser proxy;

Figure 5 is a block diagram illustrating the dataflow accompanying the use of third party assertions;

Figure 6 is a block diagram illustrating the dataflow and network configuration for using a homesite to verify a user against an access control list;

Figure 7 is a flow chart illustrating a method of the present invention to link a GUPI to an existing account at a membersite;

Figure 8 is a block diagram illustrating the dataflow and network configuration for using the account linker module of the present invention;

Figure 9 is a flow chart illustrating a method of obtaining an authenticated user approval of a privacy policy;

Figure 10 is a flow chart illustrating a method of providing authenticated user approval of a privacy policy;

Figure 11 is a block diagram illustrating the use of multiple homesites;

- 6 -

Figure 12 illustrates a site security system of the present invention; and

Figure 13 illustrates a homesite appliance of the present invention.

DETAILED DESCRIPTION

Generally, the present invention provides a method and system for identity management. Though described herein in relation to a distributed hierarchical identity management system, one skilled in the art will appreciate that it is not always necessary that the identity management system be both distributed and hierarchical.

Figure 1 illustrates a general overview of an identity management system 100 of the present invention. In a distributed hierarchical identity management system, such as the one illustrated in Figure 1, users are assigned a global unique identifier (GUID), which is also referred to as a globally unique persona identifier (GUPI). The identity management system 100 is administered by root 102, which ensures that a homesite, such as HS1 104 and HS2 106, is authoritative for any given globally unique persona identifier (GUPI) associated with a user. A membersite, such as MS 108, provides the user a service, and requires identifying information about the user that will be obtained from HS1 104. Simple site server (SSS) 110 is not used in this particular scenario, nor is HS2 106, but will be described in more detail below. A user employs browser 112 and optionally browser proxy 114.

Browser 112 initiates a session with MS 108, and is then asked for identifying information, or a login. The user indicates, via datapath A, that the required information can be provided by HS1 104. MS 108 generates a request for the information stored by HS1 104, and forwards it, preferably as an https form, to the user, via datapath B. The https form preferably has a nonce associated with it, to act as a session identifier, and includes a redirect to HS1 104, via datapath C. If the user is not already logged in and authenticated, HS1 104 obtains login information to authenticate the user. The login information can be obtained in a number of ways, including through obtaining a username and password pair, receiving a secure token, or through an out-of-band communication as discussed below. The user authentication process is not required to use http, or other standard protocols, though in a presently preferred embodiment the authentication does use a username and password combination provided over an https channel. HS1 104 then preferably seeks user authorization to release the requested information to MS 108.

- 7 -

The requested information could be as simple as the GUPI provided as a URL encoded name-value pair, or as rich as an XML fragment. Upon receiving the user authorization to release the requested information to MS **108**, HS1 **104** sends an https form to the user along datapath D. The form sent on datapath D contains a redirect to MS **108** and causes an https post, over datapath E, to deliver the requested information to MS **108** along with the nonce so that MS can properly identify the received information. One skilled in the art will appreciate how to generate the redirects, which when automated provide the user with a seamless experience.

Conventionally a get can be redirected, but a post cannot be. Posts allow a homesite to provide a membersite with user information in response to the initially transmitted form. If a user has indicated that they would like a seamless login from clicking a membersite login link, so that they are simply redirected to the homesite, and then bounced back to the membersite without having to acknowledge the homesite, the homesite must redirect a post to the membersite. Conventionally, the redirection of a post results in a user alert, impairing the user's seamless experience. To accomplish a redirected post, HS1 **104** sends browser **112** a form page with login or user data preferably stored as a hidden field. The use of the hidden field prevents a page from being loaded on browser **112** containing data that may confuse the user, and will detract from the appearance of a seamless login. The use of the hidden field is optional. This page preferably contains a link to MS **108** that when followed will post the hidden data to MS **108**. Conventionally, the user would be asked to click on the link to post the data to MS **109**. Through the use of a javascript™ command, or other such scripting language command, on the webpage, the link can be followed automatically, without showing the user what is happening. If the user has javascript™ disabled, the browser **112** simply displays a link indicating that the user should click the link to continue. With the redirected post, any node in the identity management network **100** can redirect a user to another node using a post to push information along with the user. In the following description, any post described can be replaced with the above described redirected post.

One skilled in the art will appreciate that user information can be provided in an XML block or fragment. To protect the user's privacy, and security, the GUPI, nonce and timestamp are often signed by the HS to allow for authentication of the information, and passed through an encrypted channel such as a secure sockets layer (SSL) or hyper-text

- 8 -

transfer protocol secure sockets (https) channel. However, at this point MS **108** has a statement from an entity it may have never encountered before, HS1 **104**, that a user has certain attributes, including a particular GUPI. MS **108** needs a mechanism to verify that the information is correct, or at least can be trusted to a certain degree. It may be possible to determine that HS1 **104** is part of the network **100**, and then create a back channel to HS1 **104** to obtain the certificate, herein after interchangeably referred to as a cert, to determine if the data has been properly signed. One skilled in the art will appreciate that for a variety of reasons, including certificate expiration, HS1 **104** will occasionally change the certificate or key with which the signature is performed. These changes in certificate make administration of a key ring very complex. The establishment of a back channel requires that MS **108** and HS1 **104** operate in a trusted environment, and that the back channel initiation cannot be accomplished by another party. These conditions are not always possible to maintain. By participating in the identity management network **100**, MS **108** is expressing trust in root **102**. Root **102** participates in registering a user with given homesites, such as HS1 **104**, and can provide an assertion to HS1 **104** that it is authoritative to authenticate a user as being associated with a GUPI. This assertion preferably takes the form of an XML fragment containing the GUPI and a homesite identifier, such as a homesite identification tag or the homesite cert id, also referred to as the homesite key id. This XML fragment is then signed by root **102** and provided to HS1 **104**. When HS1 **104** provides a response to MS **108** through browser **112**, it can sign the response, and include the root signed assertion. MS **108**, can verify the root signed assertion, and see that HS1 is in fact authoritative for the provided GUPI, and can also verify the certificate used by HS1 **104** to sign the response. The response can then be verified as being signed by HS1 **104**. Thus, MS can verify the data as being asserted by HS1 **104**, and can further verify that root **102** has delegated authority to HS1 **104** to make assertions for the provided GUPI. Thus, based on a trust relationship with the centralized root **102**, MS **108** can interact with HS1 **104** without having a pre-existing trust relationship. Further assertions and delegations of authority can be performed and will be discussed in more detail below.

As discussed above, when a user signs up at a homesite (HS), the HS receives a root-signed datablock containing the GUPI and the HS identifier. The HS receives one such datablock for each GUPI it is authoritative for. Upon receiving a user, a MS can

- 9 -

guess the HS associated with the user by examining a cookie provided by the HS. If an HS sets a cookie for a user that it is not authoritative for, the user can determine that they have been routed to the wrong HS, either before or after being redirected for validation. In the event that the user does not notice that they are at the wrong HS, a HS that is not authoritative for a GUPI cannot provide the datablock signed by the root showing that the HS is authoritative for the GUPI.

When MS **108** attempts to obtain authentication of the user, typically through redirection of browser **112**, MS **108** provides HS1 **104** a nonce , such as a session ID or other such tracking information which is considered to be probabilistically unique. When HS1 **104** provides authentication of the user, typically by providing MS **108** with both proof of being authoritative for the user's GUPI and the requested log in information, the nonce is typically returned with a timestamp, so that MS **108** knows when the authentication was performed and can match the response to a request. The authentication and user information can be provided through a number of channels including through a back channel between MS **108** and HS1 **104**, though in a presently preferred embodiment the user provides this information to MS on behalf of HS1 **104** through a redirection. To ensure that HS1 **104** authentication information has not been altered in transit, HS1 **104** typically signs the data block. To verify the signature and show that it is authoritative HS1 **104** transmits a datablock, including the GUPI and a HS identifier, which is signed by root **102**. The HS identifier is used to authenticate the datablock containing the user authentication information by allowing MS **108** to verify HS1 **104**'s key was used to sign the data. It is foreseen that an HS cert may not be signed directly by the root **102**, but instead may be signed by another authority, that is itself signed by root **102** (either directly, or through a series of other stages). This delegation of authority will be explained in more detail below. In a presently preferred embodiment of the invention, HS1 **104** provides the entire cert chain from HS1 **104** to root **102**, including all intervening authorities. This cert chain allows MS **108** to validate that the user information has not been tampered with, and that HS1 **104** is authoritative for the GUPI, without requiring a complex key ring, or a series of back channel communications to a number of intervening authorities.

In Figure 1, MS **108** provided the data request over an https channel. This provides a degree of security and prevents the user information from being transmitted in

the clear. However, there are a number of situations in which either MS **108** is not sophisticated enough to use a secure channel, or the data that is being requested is not sensitive enough to require the secure channel. One such example of this is that MS **108** simply wants a postal code and city of residence from a user. As this information is not being transmitted with other identifying information, there is no need to use a secure channel. However, to obtain the information from HS1 **104**, browser **112** must be authenticated. The authentication information typically includes a username and password combination, or other such identifying information that should only be sent by a secure channel. Because the request from MS **108** used standard http, the response to the request must be in standard http, though the user authentication with HS1 **104** should be done over https. If the http form results in an https post, the user will typically be presented with a warning message. This is considered to be inconvenient to the user, and results in the user not being presented with a seamless experience after clicking on a login button. To overcome this, the network **100** of the present invention provides a mechanism to allow for HS1 **104** to securely authenticate a user and provide the user information over an insecure channel, as is illustrated in Figure 2. In datapath A, browser **112** indicates that the user information is available through HS1 **104**, as before. Datapath B is used by MS **108** to send an http form to browser **112** to be forwarded to HS1 **104**. The http form causes browser **112** to send an http post to HS1 **104** over datapath C. HS1 **104** now needs to create a secure connection to browser **112** over which the user authentication can be performed. To accomplish this, HS1 **104** transmits an http response, over datapath auth1, to browser **112**. Contained in the http response transmitted over datapath auth1, is a refresh instruction causing browser **112** to refresh with an https get which is transmitted over datapath auth2. This can be transmitted as a blank page with hidden data and a standard HTML refresh command. This allows HS1 **104** to create a secure connection, preferably using https, over which browser **112** can be authenticated for a GUPI. When the user has been authenticated, and has agreed to the release of the requested information, HS1 **104** must now send the requested data to MS **108**, via browser **112**, using http instead of the https connection presently in use. To accomplish this, HS1 **104** transmits an https response to the browser over datapath auth3. This https response forces a refresh with an http get transmitted over datapath auth4. In response to the http get, HS1 **104** transmits the information requested by MS **108** to the

- 11 -

browser **112** over datapath D using http. Upon receiving the requested information, browser **112** is redirected to MS **108** and provides the requested data using an http post over channel E. Thus, MS **108** is able to request information over http and receive the response over http, while browser **112** and HS1 **104** communicate to each other using https for the authentication operations. The user of the secure channels provides both security, and gives the user some proof that HS1 **104** is the homesite that it claims to be.

To provide a mechanism to easily integrate an existing site into the identity management network **100** as a membersite **108**, simple site server (SSS) **110** can be provided. This additional server allows a vendor with an existing server to accept simple name value pairs to be provided from HS1 **104**, without verification. This allows the simple name value pairs to be provided by HS1 **104** to MS **108**, through browser **112**, and have the data to be posted directly into the existing scripts at MS **108**. Figure 3 illustrates the architecture and sample data flows of the present invention.

Browser **112** is launched and loads a page at MS **108**, as demonstrated by dataflow A. MS **108** is an existing server designed to receive information using a standard HTML form. The page generated by MS **108** has been modified to include a link to allow a user of identity management network **100** to use HS1 **104** to provide the information requested by MS **108**. Whereas MS **108** would normally guess the homesite based on a cookie carried by browser **112**, the link forwards browser **112** to SSS **110**, as shown in dataflow B'. In one presently preferred embodiment, SSS **110** is customized around MS **108** and the data that it requests from the user, in an alternate embodiment, MS **108** instructs SSS **110** that the data should be provided as a set of name value pairs, and provides the name-value pairings. SSS **110** guesses the user's homesite, much as MS **108** did in Figure 1, and provides the data request to browser **112** over datapath B. Browser **112** is then redirected to HS1 **104** over datapath C. This data request also includes a return address, so that HS1 **104** can forward the data directly to MS **108**, and not SSS **110**. Browser **112** connects to HS1 **104** and provides the name request from MS **108** over datapath C. HS1 **104** authenticates the user and obtains authorization to release the requested information. Upon receipt of the authorization, HS1 **104** provides the requested data, formatted as the name value pairs defined in the request from MS **108**, to the browser **112** over data path D. Browser **112** is then redirected to MS **108** over datapath E. The data sent through browser **112** by HS1 **104** is provided to MS **108**, and is

- 12 -

posted directly into the form processing scripts. Thus, MS **108** is able to request information from HS1 **104** without requiring a rewrite of the scripts, through the simple addition of a networked computing appliance, that can be provided as a discrete hardware component custom designed and optimized for its task.

One skilled in the art will appreciate that SSS **110** functions as a proxy to MS **108** for communication to the identity management network **100**, and in particular to HS1 **104**. The link that a user clicks upon to indicate membership in the identity management network posts a request that would normally have been directed to HS1 **104**. Instead this post is made to SSS **110**, which acts as a proxy and guesses the homesite, redirects the user to the guessed homesite. From this point in the process, the same dataflows used in the previously described methods are followed.

For numerous thin clients, such as smart phones and personal digital assistants with data access, bandwidth and connection speed are a limited. If assertions are signed and transmitted from the homesite to the thin client browser, there can be noticeable slowdown related to the transfer of these assertions. Additionally, these clients typically have data connections that are metered, resulting in additional cost to transfer the assertions from the homesite to the browser and then back to the membersite. To provide a mechanism to facilitate reducing the data transmitting in this fashion, an embodiment of the identity management network **100** of the present invention takes advantage of a trust relationship between the user and a proxy service, typically offered by the cellular carrier providing the data service. A network model of this embodiment of the present invention is illustrated in Figure 4. All interaction between browser **112** and the rest of the network **100** is routed through browser proxy **114**. When the browser **112** initiates a session with MS **108**, Browser **112** instructs browser proxy **114** over datapath A' to create a session with MS **108** over data path A. Browser Proxy then receives and receives the redirection command over datapath B. Browser **112** and browser proxy **114** are redirected to HS1 **104** data path C, where the user authenticates using datapath C' to provide browser proxy **114** with the authentication information and to approve the release of the requested information. Upon authentication, HS1 **104** sends the data to browser proxy **114** over datapath D. This information is redirected to MS **108**, over datapath E, bypassing browser **112**. This allows browser proxy **114**, which has a faster network connection than browser **112**, to minimize the data sent over the slow connection to browser **112**. Because browser

- 13 -

112 has an existing trust relationship with the cellular carrier, or other such entity, operating browser proxy **114**, browser **112** is able to delegate activities to browser proxy **114** to increase the effective speed of the system. It is recognized that due to the trusted relationship between the user and the cellular carrier, the browser proxy **114** and HS1 **104** may be operated by the same entity, and may even be integrated for ease of network planning. It is also recognized that this proxy setup need not be limited to thin clients, but may be employed by other devices, such as standard personal computers with dial-up or metered access who wish to reduce the amount of data transferred to and from their computer.

HS1 **104** is determined to be authoritative for the authentication of a GUPI if it can provide a root signed block of data including a homesite identifier, and the GUPI. By signing this datablock, root **102** is delegating authentication responsibility for the GUPI to HS1 **104**. The idea of such a delegation of authority, and the accompanying assertion, can be combined with a chain of certificates to provide a wide variety of features.

A user is assigned a GUPI when registered into the identity management network **100**. This GUPI is, by definition, unique. The HS that registers the user is made authoritative for authenticating the user associated with the GUPI. The HS then stores a profile for the user. Parts of the profile may include information about membership in loyalty or affiliate programs, and the status of the membership. For example, a loyalty program may provide a signed assertion to HS1 **104** that the user associated with a particular GUPI has a particular membership number, and has reached a particular level in the loyalty program. This assertion can be broken into two elements, the first element being the user's membership number the second being the membership level. The first element is permanent, while the second element is time sensitive. Thus, the loyalty program, the authoritative site AS **140**, can provide the user with two signed assertions. The first assertion is the membership number, while the second is the membership level. The second assertion can be provided with an expiry date. When MS **108** requests the user information, including any information pertaining to the loyalty program, HS1 **104** will assemble the subset of the user's profile to be sent, and in the process will preferably check to see if the assertions are still valid. If an assertion by a third party is out of date, HS1 **104** can contact the AS **140** and obtain a new assertion. This updating of a stale assertion can be completely performed in a back channel without user involvement, or can

- 14 -

be performed by re-routing the user to the loyalty program to obtain the information and bring it back to the homesite.

Figure 5 illustrates how, in one embodiment, assertions are obtained and provided. Browser **112** initiates a session with an authoritative site **140** and provides identifying information in datapath assert 1. This information can include a shared secret, or may be provided by physical presence at AS **140** and use of a specified browser **112**. AS **140** obtains at least one GUPI associated with the user of browser **112** (which can be obtained from HS1 **104** by authenticating), and creates assertions containing asserted statements which can be strings or XML fragments, and the GUPI to which the assertion is made. These assertions are signed and sent, via browser **112**, to HS1 **104** along datapath assert 2. Browser **112** later initiates a session with MS **108** along data path A. MS **108** requires the previously obtained assertion, and likely wants user login information as well. As a result, MS **108** sends a login request to browser **112** along datapath B, which redirects browser **112** to HS1 **104**. Authentication of the user at HS1 **104** is performed along datapath C, and permission to release the assertions is obtained. HS1 **104** preferably examines the requested assertions to determine if they have expired. Preferably each assertion that expires includes a refresh element that dictates how an updated or refreshed assertion can be obtained. In one embodiment, the updated assertion is obtained by surrendering the stale assertion via a back channel specified in the refresh URI. The refreshed assertion is obtained by HS1 **104** from AS **140** along datapath assert refresh. The assertions and user information, are sent to browser **112** along datapath D, and are then redirected to MS **108** along datapath E.

Third party assertions can be signed by the asserting party AS **140**, but a chain of trust must exist between the third party and MS **108** for MS **108** to understand, or trust, the assertion. If a series of assertions are provided by a university (AS **140**) indicating that a user associated with a GUPI has a particular student number, is enrolled in an engineering program, and is a student in good standing, each of these assertions would be signed by the university. However, there are a large number of universities, and requiring MS **108** to store all the keys for each university would result in a very large key ring to maintain. To overcome this problem, a university (AS **110**) can prove that it is a bona fide post secondary educational institution by providing an assertion from an accreditation body that it is a recognized institution. The university's assertions about a

- 15 -

user would be stored in the profile of a GUPI at HS1 **104**, and would include the accreditation body assertion that AS **140** is delegated responsibility to make assertions. Thus, MS **108** can simply recognize the accreditation body, and determine from the accreditation body's signed assertion that the university is accredited. The accreditation body's signed assertion would include the university's public certificate, or public certificate id, so that the signed assertions by the university can be recognized. This multi-tiered system of assertions allows MS **108** to reduce the key management issues, and still support a large number of third party assertions.

Following the university example outlined above, the assertion that the user of browser **112** has a particular student number would not expire, the assertion that the user is enrolled in an engineering program would expire at the end of an academic session, and the assertion that the user is a student in good standing could expire daily. When a user visits the university library website (MS **108**), and uses the identity management network to log in, the three above described assertions are requested, so that the library (MS **108**) can confirm that the user of browser **112** has access to the general collection (based on student number), specific collections related to the engineering faculty, and is allowed access because the student is in good standing. However, when requested for the assertions, HS1 **104** checks their expiry dates. The daily expiry of the in good standing assertion requires that the user of browser **112** authorize HS1 **104** to update the assertion (though authorization can be provided once allowing HS1 **104** to automatically refresh the assertion). Expiring assertions preferably include a refresh element containing a URI to indicate where and how the assertion can be renewed. This URI is preferably used by HS1 **104** to obtain an updated assertion that the user is a student in good standing from AS **110**. Upon obtaining the updated assertion, the user information, including the assertions, is provided to browser **112** which is then redirected to MS **108**, which in this case is the university library.

HS1 **104** can provide a number of information release options to the user. When assertions, or other information, are changed from a previously stored value, HS1 **104** can be permitted to automatically provide the new data. Another option is for an assertion to be released only if the assertion has not changed. Other variations will be well understood by those skilled in the art, as will the implementation of such a rule processing mechanism.

MS **108**, upon receiving the user information, is provided third party assertions, including the assertion that HS1 **104** is authoritative for the GUPI. The highest level assertion in a chain delegates asserting responsibility to the next highest level entity. This entity can delegate the asserting responsibility to another entity. This continues, until one of the entities asserts a property regarding the user. This decouples the trust and assertion relationships, but provides a simple mechanism for MS **108** to be assured that the data is properly asserted, though the entity making the assertion is unknown to MS **108**.

Assertions can contain expiry dates that dictate when the assertion can expire, refresh properties that dictate how an expired assertion is updated, and delegation limits that dictate how many delegations can be made before MS **108** should not consider the assertion valid. Some entities dealing with confidential information may require that the assertions cannot be delegated more than a set number of times, so that the assertion is not misused.

Various techniques for obtaining updated assertions will be known to those skilled in the art. Such techniques include the use of a refresh URI in the assertion, and user redirection to AS **140** to re-obtain the assertion. In one embodiment, a refresh URI is provided and HS1 **104** connects to AS **140** through a back channel and provides the user's GUPI, which AS **140** compares to a local database and determines that the GUPI is associated with expired certificates, and the user associated with the GUPI has authorized their automatic release to HS1 **104**, and resultingly issues refreshed assertions. In another embodiment, HS1 **104** uses the refresh URI, provides the expired assertion and obtains a new assertion in its place. In a further embodiment, HS1 **104** redirects browser **112** to AS **140** to authenticate and authorize the release of a new set of assertions. AS **140** can have different refresh rules for each assertion that it issues, so that assertions that a student is in good standing can be refreshed without user intervention, while assertions such as enrolment in a program require user validation that the assertion should be released.

One skilled in the art will appreciate that the identity management system of the present invention allows for the building of an analog to the trust models of the real world. A user is presented to MS **108** with a statement that the user is a particular individual. This statement is accepted because it comes either directly from a trusted source, or from

someone who is delegated the responsibility to make the statement by a trusted source (HS1 **104** is delegated authentication authority by root **102**). Third parties can make assertions about the user. These assertions can be requested by MS **108**, much as proof of membership in a program can be requested in the real world. However, due to its digital nature, the identity management system of the present invention can provide a higher degree of granularity for the information. While in the real world, a user may be able to obtain a discount upon proving that he is a student, this is accomplished by providing a student card, which provides a student number. In the identity management system, an assertion that a user is a student can be made without providing the student number. A user could also provide an assertion that they live in a particular city, without providing an address, whereas in the real world such an assertion would be validated by providing identification which typically contains a full address. As a result, the identity management system **100** of the present invention provides many analogs to the real world, but also offers a much higher degree of granularity to the information released.

It is common for websites, such as a personal photosites, to restrict access to a set of individuals, largely for privacy concerns. At present, this is commonly done through an obfuscated uniform resource locator (URL), under the assumption that because the URL is not published and is difficult to guess, only people who should know about the site will know where to go. Other more sophisticated systems include a username/password protection system, but this typically requires more system administration than is desirable for a small site, as each user would have to be authenticated prior to issuing a username/password pair. A solution to this is the use of an access control list (ACL) that stores identifying information, such as an email address. When a user attempts to access the photosite he or she is requested to provide an email address that is checked against the contents of the ACL. If the address is present in the ACL then the use is given access. To protect the user email addresses that are stored in the ACL it is common for the contents of the ACL to be stored solely as a hash, so that a malicious intruder cannot simply abscond with a copy of the ACL and determine who has access to the site. This solution, though simple, does not authenticate that the user is who he or she asserts to be, it simply tests that the user knows that a particular individual has access to the site. Figure 6 illustrates a network configuration allowing a content site **118**, such as the photosite, to employ the identity management network of the present invention. The login

to the photosite **118** can make use of the identity management system by having the login managed by MS **108**, a membersite in the identity management network **100**. Upon visiting the photosite **116**, the browser **112** indicates that its identifying information can be provided by HS1 **104**, using datapath A. This is typically accomplished by the user clicking on a link, but can be done in any number of other ways including the reading of a cookie indicating that the user prefers to use the identity management system **100**. Photosite **116** redirects browser **112** to MS **108** over datapath A'. MS **108** guesses the correct HS, and redirects browser **112** to HS1 **104** over datapath B. The information request from MS **108** to HS1 **104** includes the address of photosite **116** as a return address. Upon authentication of the user, HS1 **104** issues identifying information to the browser **112** which is forwarded to photosite **116** over datapath E. The identifying information is passed over datapath E to photosite **116** which can compare the information to the contents of the access control list **118**.. Often the information provided by HS1 **104** is the user's email address, which may be hashed prior to being transmitted. By transmitting the user's email address as a hash, the identity of the user is protected against a third party attempting to eavesdrop on an unencrypted channel. One skilled in the art will appreciate that this reduces the administrative overhead of maintaining an ACL **120**, and allows a simplified interface to be provided to a photosite operator to add to and delete individuals from the ACL **120**.

Where prior art identity management systems do not provide a migration path from existing user bases to those supported by the identity management system, the present invention provides an account linker module that can be deployed in conjunction with MS **108** to allow an existing user base to be associated with GUPIs provided by the identity management system. Figure 7 is a flowchart illustrating an embodiment of the process used by the account linker module, and is best understood in conjunction with Figure 8 which illustrates a network overview of the account linker module.

Browser **112** initiates a session with MS **108**, and indicates membership in the identity management system, as shown in data flow A. This indication of membership is typically performed by clicking on a link. This link redirects the user from MS **108** to the account linker **120**, as shown by dataflow A'. Upon receipt of the redirected user, in step **150** account linker **120** guesses the user's homesite and redirects the user to HS1 **104** for authentication, and optionally to obtain data, as shown in data flow B. Upon authentication

- 19 -

of the user at HS **104** (not shown), the user is redirected to account linker **120** in step **152** as shown by data flow E. Account linker **120** optionally authenticates the authoritativeness of HS1 **104** for the supplied GUPI in step **156**. Account linker **120** then examines the GUPI and compares it to a concordance table linking GUPIs and existing account information, as shown in step **158**. Preferably this information is stored in the user account database **122**, this communication is illustrated in and dataflows F and F'. If the GUPI has been seen before, it will have been linked to an account in account database **122**, provided in data flow F', and the concordant account is used by the user to login in step **160**, as shown by data flow G. If the GUPI has not been seen before, account linker **120**asks the user if he or she has an existing account in step **162**. If the user indicates that there is an existing account, the account is linked to the GUPI in the account database **122**, as shown in step **164**, and in the future, when the GUPI is received it will be linked to the user account. If the user indicates in step **166** that a new account should be created, the new account is created in account database **122** and linked to the GUPI. After either linking the new GUPI to an account, or creating an account linked to the GUPI, the account linker **120** passes browser **112** back to MS **108** for login, as shown in step **168** and dataflow G. The passing of browser **112** to Ms **108** for a login preferably is performed by account linker **120** sending a redirected post to browser **112**. Browser **112** is then redirected to MS **108** and the userid and password combination associated with the GUPI in the account database **122** is provided, allowing browser **112** to bypass the login screen. MS **108** can then check account database **122** to verify the information, or can be passed a verification message along with the login request that can be verified by account linker **122**.

Because the user is often releasing personal information that is protected by privacy policies, it may be important for the membersite to obtain user permission to release the information. Presently, to obtain user permission to release data the membersite would have to show the user a privacy policy, and ask the user to click a link to acknowledge agreement with the policy. Thus, if the membersite changes the policy they have no mechanism to show that the user agreed to the new policy or the old policy other than the date at which the data was provided. The user can also contest that new terms were added to the policy that were not agreed to in advance, as there is no reliable audit trail. In a present embodiment of the network of the present invention, permission

can be obtained by providing the user a copy of a privacy policy, such as a Platform for Privacy Preference Project (P3P) compliant policy or a privacy statement, and asking that the user acknowledge the policy. It may also be important for the membersite to keep an audit trail showing that the user agreed to release the information. The network of the present invention provides a mechanism for doing so. When MS **108** redirects browser **112** to HS1 **104**, it can provide, as part of the redirection, a privacy policy, as shown in step **170**. The privacy policy is received by HS1 **104** in step **178**, and can be text to be displayed by HS1 **104** outlining the terms under which the user information will be released, it can be a universal resource indicator (URI) pointing to a privacy policy complying with the P3P standards, or it can be provided in other user displayable formats that will be understood by those skilled in the art. Prior to obtaining user permission to release the requested information, HS1 **104** provides the policy to the user in step **180**. The policy can be displayed in full, or a machine comparison of the policy to user specified rules can provide the user with a synopsis of the differences between the user-specified rules and the policy. HS1 **104** then obtains user approval for the release of the user data in step **184**, preferably after authenticating the user in step **182**. One skilled in the art will appreciate that user authentication can occur prior to displaying the policy in step **180**. Prior to HS1 **104** redirects browser **112** to MS **108** in step **188**, it signs the datablock returned to MS **108** containing the user information and the privacy policy in step **186**. In different embodiments, the user information and the privacy policy can be separately signed, or as implemented in the presently preferred embodiment, they can be contained in the same XML fragment, which is then signed by HS1 **104**. This signed XML fragment contains the GUPI of the user, the user's information and the privacy policy and is received by MS **108** in step **172**. Thus, MS **108** is able to determine that HS1 **104** is authoritative for the GUPI, by looking for an assertion by root **102**, and then can determine that HS1 **104** signed the privacy policy on behalf of the user in step **174**. This signed XML fragment can be kept by MS **108**, as shown in step **176**, to ensure that if future questions are raised, it can be proved that the user, through HS1 **104**, authorized the release of the data under the privacy policy.

One skilled in the art will appreciate that a single user may elect to obtain a number of GUPI's for different online personas. Anywhere from one, to all of the GUPI's held by a user can be associated with a single HS. Thus, for multiple GUPI's, a user may

- 21 -

elect to use a plurality of different HS's. Conversely, it may be advantageous for a use to make a plurality of HS's authoritative for a single GUPI. This allows for redundancy in the event that a HS is unavailable for technical or business reasons. It also allows for a user to provide only a subset of any information to a single HS, and use the different HS's in series to hold a complete profile. This is illustrated in Figure 10, where both HS1 **104** and HS2 **106** are authoritative for a single GUPI associated with the user of browser **112**. As the only evidence that either HS requires to prove that it is authoritative is a datablock containing the GUPI, and the HS identifier (hereinafter depicted as [GUPI, HSid]), signed by the root (signatures of a datablock herein after depicted as a superscript to the signed datablock, e.g. [GUPI, HSid]^{root}), both HS1 **104** and HS2 **106** can be made authoritative for a single GUPI. To initiate a plurality of HS's as being authoritative for a single GUPI, a user creates a first profile with HS1 **104**. HS1 **104** collects information from the user and stores the data. Upon authenticating that the user wishes to associate his or her GUPI with HS1 **104**, root **102** provides [GUPI, HSid(1)]^{root} to HS1 **104**. At a later time, the user visits HS2 **106**, and signs up using the same unique email address as was provided to HS1 **104**. Root **102**, upon determining that there is already an authoritative entity for the provided email address, asks the user, preferably through an email, what to do. Typically the options are provided to the user as a series of embedded links in the email message. One of the options is to ignore the request, which simply terminates the scenario. Another option allows the user to revoke HS1 **104**'s authoritative status, and designate HS2 **106** as the authoritative entity. In this case root instructs HS1 **104** to get rid of [GUPI, HSid(1)]^{root} , and provides to HS2 **106** [GUPI, HSid(2)]^{root}. In the event that the user selects to add HS2 **106** to the list of authoritative HS's, HS2 **106** is provided [GUPI, HSid(2)]^{root}. To allow the user to bypass the manual entry of data, root **102** initiates a session with HS1 **104** acting as a MS, and redirects the user to HS1 **104** to authenticate him or herself. This authentication allows root **102**, while acting as a MS, to request all the information associated with the user, and provide that information to HS2 **106**. In an alternate embodiment, HS2 **106** can act as a MS, and request that the user authenticate himself with HS1 **104** to get access to the profile stored by HS1 **104**. Allowing multiple HS's to be authoritative for a single GUPI provides the user redundancy in the event that one of the HS's becomes unavailable, and additionally allows the user to trust one subset of the profile to one HS, and another subset of the profile to another HS. When the user

- 22 -

visits a MS, there are a number of different mechanisms that can be used to select which HS to use. If there is only one HS indicated by a cookie, the HS indicated by that cookie would become the default HS. Alternately, if a plurality of HS's are indicated by a cookie or a series of cookies, the MS can provide the user with a drop down list of the HS's indicated. If there is no cookie, or as a further option, MS can prompt the user to provide a HS name, such as homesite1.com, which is then mapped to the shadow domain name of homesite1.com.root.net if a shadow domain administered by the root is being employed to judge membership in the identity management system.

As multiple HS can be authoritative for a single GUPI, and the entire cert chain is provided to MS, a variety of different platforms can be used for a HS. One such platform is a locally executed service designed to hold data associated with a single user. A user with multiple devices could employ a multitude of different local HS's, so that a laptop computer would have an HS, as would a connectivity enabled cellular phone. An Internet based HS can serve as a backup, a fall back when not using the designated devices, or as a pivot point to synchronize data among the different local HS. As the user validation process relies upon an agreed upon method decided by the user and the HS, a locally executed application could rely solely upon the operating system login process to validate a user. All requests from a MS would be routed to the user's system and served by the locally executed application which would be set to receive requests at a defined port. In another embodiment, the local HS can reside on a peripheral device, such as a USB memory fob, or a Bluetooth enabled memory device. These devices can be portable, and include mechanisms such as Radio Frequency Identification (RFID) tags, or biometric readers to assist in the user authentication process. These devices could thus function in other environments too.

For a locally executed HS, the cookie would allow MS to perform a DNS request on a unique name such as localwallet.root.net, whereas a conventional HS would refer to homesite.com.root.net. The DNS entry for localwallet.root.net, or another such entry, are preferably mapped to either the CNAME value 'localhost' or to a loopback address. A search performed on cmd.localwallet.root.net would preferably result in a TXT entry of http://localhost:1234/commandscript, where 1234 is a predetermined port on which the locally executed HS listens, and commandscript is a script used to receive and transmit information to MS. Such a local HS need not rely upon the creation of a web based login

process, and can use any number of options, including the use of a locally executed application to obtain identifying information, or the reliance upon the user's login to the computer used.

As the management of the root.net domain allows for a near infinite number of subdomains, a plurality of different local HS products can be sold. Each of the local HS's can then be assigned any non-top level domain base name, to allow co-existence of a plurality of different local HS's, each of which preferably use a different port to allow their co-existence on a single computer. Two users with the same local HS application, would both be using the same DNS entry, but sent to different local computers. To address privacy concerns each local HS provider can use a plurality of different HS certs, so that preferably no two local HS's use the same cert. The local HS cert is then signed by the local HS vendor, whose cert is signed by the root. This allows MS to validate the user information through the validation of the cert chain. If the vendor cert is time limited and is provided with an expiry date, the GUPI can be associated with a unique serial number attached to the local HS, so that if a physical device containing the HS is reported lost, the lost HS can only be used until the HS vendor cert expires. In another embodiment, the near infinite domain space in an identity management network shadow domain can be employed to assign each locally executed HS with its own DNS entry, preferably assigned during the initiation of the locally executed HS. If such a HS is lost, the domain can be removed from the DNS system, providing deactivation of the HS in short order.

Version control of functionality associated with the identity management system can be provided through the use of the DNS entries associated with the HS. If there are two versions of the identity management system, assuming for the sake of simplicity that they are backwards compatible, a MS supporting both version 1 and version 2, upon receiving indication from the user that they want to be authenticated by HS_alpha.com, can perform a DNS lookup on cmd2.hs_alpha.com.root.net to determine if version 2 is supported. If the DNS lookup fails, MS can perform the DNS lookup of cmd1.hs_alpha.com.root.net to determine which address the user should be redirected to. The difference in versions can be used to support a richer data schema, a differently organised data schema, and to implement novel features. If a particular feature is provided by a subset of HS's, and is considered to be version independent, a DNS entry mapping feature.HS_alpha.com.root.net to a valid address can be provided to indicate

that the desired features is supported. In an alternate embodiment, the HS identifying cookie can indicate the features and versions supported to MS.

Alternately, an HS can transmit as a property of the XML document a date associated with the schema used to store the user's data. If an MS receives a response that does not include the requested information, and the schema date is earlier than the date on which the field for the data was added to the schema, it can be determined that the user does not have that information stored at the HS. This would allow the membersite to recognize that the information was not stored, and allows the membersite ask the user for the information, without worrying that the data was not provided by choice.

It may be desirable to provide an alternate method of authenticating a user at a HS. Typically an HS will store, as part of the user profile, contact information which can include a telephone number. An HS can allow for out of band authentication of a user through the use of a system that would call a user at a designated number, such as a cellular phone number, and request proof of authentication. One such type of proof of authentication is simple possession, on the assumption that if a user designates a phone number as authoritative and secure then other parties do not have access to the number without permission. In the case of a cellular phone, a lost phone can be reported stolen and all calls to it would be disabled, and authentication over that channel would be disabled. In an alternate embodiment, the user would receive an automated call and be required to provide a numeric sequence, such as a personal identification number (PIN), using the dual-tone-multiple-frequency (DTMF) telephone keypad. One skilled in the art will appreciate that using such a mechanism, the HS could additionally prompt the user for the telephone number to be reached at, so that while travelling, or simply at a location away from the designated phone, another phone number can be used. In a further embodiment, the out of band authentication requires information other than a PIN, such as a request for a birthdate, street address of a familiar location, or other such information. The HS can select which question the user is asked from a list of provided questions and answers, so that there is a higher likelihood that the user has in fact be asked information that only he or she knows. In a further embodiment, the out of band authentication can be a telephone call from a live operator asking for detailed information, such as a mother's maiden name, or a pet's name. The out of band connection, however it is established,

- 25 -

preferably performs authentication through the use of a shared secret that is validated during the exchange.

To facilitate synchronizing data between homesites, each field in the schema used to store user information can be provided with a corresponding serial number or date, which is updated when information is changed. Alternatively each entry can be timestamped. The user of either serial numbers of timestamps allows homesites to determine which entry corresponds to the newest data, and synchronize using the newest data. It also allows a membersite to request an update of information by sending either a serial number or a timestamp along with a request for any information newer than the transmitted value. One skilled in the art will appreciate that though there are many different known methods used to synchronize data, none of these techniques are traditionally employed to synchronize between a plurality of homesites.

A user having visited MS **108** and provided data through authentication with HS1 **104**, can instruct HS1 **104** that the data provided to MS **108** be kept up to date. To facilitate this update, MS **108** can send HS1 **104** a request for updated information along with the authentication request sent in accordance with another login. This request for updated information can include the date on which the information was last provided. HS1 **104** can then determine whether or not the information has been updated. If the information has been updated, the user can be prompted to provide permission for the release of the updated information, or the user can have previously provided permission to release updated information whenever appropriate.

To facilitate these requests for updated information, MS **108** can provide the request as an XML query, and receive an XML reply. This request can be made at the time of a user login, so that a user visiting an e-tailer with whom he has already established an account will be directed to HS1 **104**, along with a request indicating that all information newer than a given timestamp, or more recent than a provided serial number, be provided in the response. In an alternate embodiment, MS can provide the HS with a hash of currently stored information, and HS1 **104** will transmit the user information only if the stored information does not correspond to the hash, and permission has been obtained from the user. Thus, if a user changes addresses, at the first visit to an e-tailer after moving, HS will provide the updated contact information, preferably after confirming

- 26 -

transmission of the data with the user, to the e-tailer so that that user's new information is propagated with the user's browsing habits.

Each HS can support a plurality of different authentication mechanisms for a single user, so for two HS, each authoritative for the same GUPI, the user could potentially be authenticated differently by each HS. In the event that there are two GUPI hosted by a single HS, each of the GUPI could be associated with a different authentication.

The support for different authentication mechanisms allows the user profile to reflect how the user authenticated. Thus, different authentication methods can be offered for a single GUPI. Certain membersites, including but not limited to banking and financial sites could then require that the user authenticate with a high security authentication before a login is permitted to the membersite. Thus, a homesite could offer a use the ability to login using a simple username password combination, a username password combination with certain qualifications on the password including length and the existence of special characters or case sensitivity, or a secure identifier string generated by a hardware device such as a conventional security fob. This would allow varying degrees of security to be associated with different parts of a user persona, linked to different GUPIs. Thus, different authentication mechanisms allow a user to use a "light" security login for low security tasks, such as logging into news sites, but have more robust authentication for logging into intranets and performing other tasks such as online banking. The different degrees of security attributed to each authentication can also optionally be accredited by a third party which can then provide an accreditation to a homesite based on the relative security of the user authentication mechanism. Such an accreditation can be sent to a membersite along with the user information, to certify that a secure login was performed.

Sending information between the HS and MS after user authentication can take place through either a backchannel or through sending the information through the user. When the user visits MS **108** an SSL connection is established. When redirected to HS1 **104** another SSL connection is established. One skilled in the art will appreciate that there is a time and computational overhead associated with the establishment of an SSL connection, and that these connections are persistent in a time limited fashion. With redirection between MS **108** and HS1 **104**, the already established SSL connections can be used so long as the timeout period for the SSL connection has not expired. Thus, if HS sends the information to MS via user redirection no new SSL connection is required.

- 27 -

However if HS1 **104** uses an SSL backchannel to MS **108** a new SSL link is required. This introduces additional overhead. In addition to the overhead, MS **108** needs a mechanism to authenticate that the new backchannel connection is with the homesite associated with the user that a nonce was assigned to. This introduces a possible security concern, as MS must secure another open port to prevent a malicious third party from poking around.

If a user, after establishing the SSL connections to MS **108** and HS1 **104**, visits a second MS, referred to as MS2, there is likely still an SSL connection to HS1 **104**, so there is no need for the re-establishment of the SSL link to HS1 **104**. In the worst case scenario, where no authentication occurs within the lifespan of the SSL link to an MS, the use of user redirection has the same overhead as the use of a backchannel, but for each HS to MS backchannel avoided, there is the possibility of one fewer SSL connection required.

One question that has arisen is the question of how a user can authenticate that he has been redirected to the correct HS. One skilled in the art will appreciate that Figure 11 includes prior art verification mechanisms, and an additional security feature. When a user connects to a secure server the address, preceded by 'https://' is displayed at **124**, and when the secure channel is established a lock **128** is displayed by the browser. However, the lock merely indicates that the channel is secure, and relies upon the user to examine the address **124**. Most users give only a furtive glace at **124**, and thus miss misspelled addresses. As a result, a malicious site, impersonating a MS could redirect the user to a fake HS that is mocked up to appear like the user selected HS. So long as the fake HS has a similar domain name and runs a secure server, many users will end up giving up their identifying information. The present invention provides a mechanism for providing user specified content **126** on the front page of the HS that will be difficult to reliably mock up for a fake HS. Though user specified content in the form of an image selected by the user from a list has previously been taught by PassMark Inc., other user specified information can be utilized to make a site more resistant to impersonation. User defined content **126**, such as a user defined pass phrase that the user has created, that the HS displays upon reading the HS cookie from the user is more difficult to fake than an image selected from a list. Images selected from a list can be simply rotated between, or a single image can be used, and a certain percentage of users will end up with the image that they had selected. However, user created content, such as a pass phrase, is far more

- 28 -

difficult to fake. A fake HS would not have access to the cookie, and would thus not be able to determine the identifier linked to the pass phrase. One skilled in the art will appreciate that the entire pass phrase, or a simple numeric identifier that the HS can link to the pass phrase in a lookup table would enable such a user specified display.

In an alternate embodiment, another form of user defined content 126 is provided through the use of a browser plugin. It is common for secure websites to provide an identifying logo on their front page. This logo cannot be relied upon to indicate the correct site however, as any user who establishes the connection can get a copy of the logo for placement on a fake page. However, the logo can be rendered by a plugin, or supported natively by a browser, and be provided in a new graphical image format, as an extension to an existing format, or as another data type including an XML file referencing both the image and other data. Included in the image file is a reference to the URL 124 that should be displayed on the page showing the logo. During the rendering process, the browser or plugin compares URL 124 to the URL provided by the logo, and if they match provides a user defined behavior, this behavior can either supplement or serve as the user defined content 126. This behavior may include a special visual rendering, such as a cycling of the colours in the logo and causing the colour intensity to increase and decrease, the cue may include the playing of an audio file specified by the user, and the cue may include activation of force-feedback through an input device such as a mouse, so that when the mouse is pointed to the logo it vibrates. As this cue is defined on a user by user basis, it makes creating a fake website considerably more difficult. To prevent a malicious user from carefully rendering the logo and attaching a different URL to it, the browser or plugin can be designed to provide the user defined behavior, if and only if the logo is signed by a trusted authority such as a known certificate authority. Thus, to impersonate a HS, or other secure website, a malicious third party would be required to get a similar domain name , run a secure server attached to the domain name so that the browser will indicate a secure connection, mock up a logo, and have a trusted certificate authority associate the logo with the domain name and sign the logo. The involvement of the trusted authority makes the impersonation of the website much more difficult.

In prior art distributed identity management systems, setting up identity stores is a difficult and time consuming process. Services, such as those offered by homesites, may be offered by internet service providers or other such entities wishing to offer these

services. However, setting up a homesite can be technically difficult, or at least undesirable, for some of the potential HS service providers, who are typically more interested in off the shelf deployments, which can easily be integrated into an existing environment. Such an off the shelf solution also allows for an HS embodiment that can address security concerns regarding user data. Furthermore, the service offered by a homesite service provider inherently includes storing private user data. Many service provides do not want to assume liability for storing such sensitive data. As the homesite grows, more personal data is stored, and the homesite becomes a bigger target for someone looking to compromise the integrity of the user data store. To maintain a low likelihood of a security breech, the system of the present invention provides for a secured network appliance offering homesite services. By purchasing a dedicated homesite server, a service provider can obtain an off the shelf tested appliance that offers a high degree of security. For the vendor of such an appliance, it may be considered feasible to assume liability for an up-to-date appliance, which is unlikely to be the case for a pure software sale. This allows liability to be shifted to the appliance vendor, who would then make available a secured and sealed appliance that does not allow for user tampering.

To securely store user data on an external storage device, long encryption keys can be used, making a brute force decryption overly difficult. However, the key store must be properly secured. Furthermore, conventional identity data stores encrypt all data using the same key to avoid having to manage a vast number of unique keys and the associated key management issues.

In an embodiment of the present invention, the modular nature of computing systems is employed to provide a scalable and easily deployable embodiment of the homesite. Many HS service provider candidates already employ modular computing systems, often referred to as computing appliances. Such an embodiment is illustrated in Figure 13. Homesite appliance **130** can be built around standard hardware and provided with network accessibility. The counterpoint to using standard hardware is that the selection of hardware can be made to provide a platform meeting the needs of a service provider. Such an appliance can provide a balance between power consumption and computing power, and the software executed by the appliance **130** can be optimized for the computing platform that appliance **130** is based on. In a presently preferred embodiment, homesite appliance does not store user profiles on a local storage device,

- 30 -

and instead stores them on a distinct storage array **132**. Storage array **132** can be network attached storage that can allow for modular additions to the system allowing the homesite service provide to grow the storage as needed.

Some users may have a concern about the security of their personal information being centrally stored by a homesite. To accommodate this concern it is possible to store the user data in encrypted form on storage array **132**. The encrypted data can then be decrypted on an as needed basis by homesite appliance **130**. Because much of the user data is transmitted over an SSL channel, the data is only decrypted inside homesite appliance **130**, which can be designed to prevent tampering using standard mechanisms. As a result, even the homesite service provider will not have access to the user data. To manage the encryption of the user data, homesite appliance **130** can include an encryption decryption engine **134**. Engine **134** can use symmetrical encryption and can store only a portion of the key, allowing the user to input a passphrase that allows the key to be completed. This splitting of the key in two segments allows the user control of when personal information is disclosed, because without the passphrase the decryption of the stored information cannot occur. Engine **134** can employ dedicated logic and circuitry to perform encryption and decryption of the data faster than a general purpose processor would. Connected to the engine **134** is request processor and authenticator **136**. Request processor and authenticator **136** authenticates the user, and processes the membersite requests for data. One skilled in the art will appreciate that processor and authenticator **136** can be implemented as software executed on a general purpose hardware platform. The software executed by processor and authenticator **136** is preferably optimized for the hardware platform used. Such an optimization allows for rapid handling of user authentication and data processing. To reduce the computational load on processor and authenticator **136**, the SSL encryption of outgoing data can be performed by a dedicated SSL encryption/decryption engine **138**. One skilled in the art will appreciate that such an engine, hereinafter referred to as an SSL engine, can be designed to rapidly encrypt and decrypt with the public key technology used in the SSL process. One skilled in the art will appreciate that the homesite appliance of the present invention can be implemented using a single generic hardware assisted encryption/decryption engine **135** to perform the function of SSL engine **138** and encryption engine **134**. In alternate embodiments, the

- 31 -

SSL engine **138** is omitted and the SSL encryption is performed by the same general purpose processor as is used for authenticator **136**.

Such an appliance **130** allows a homesite service provider to simply add a component to an existing network and be able to offer a high performance homesite. In embodiments of the appliance **130**, further variations are implemented to allow further performance benefits. In one such embodiment, when a user's data is retrieved and decrypted from the storage array, request processor and authenticator **136** can store the decrypted user data in an adjoining memory so that further requests from the user during the same session do not require retrieving and decrypting data from an external source. In another embodiment, the modular nature of the homesite appliance **130** is used to create a homesite, such as HS1 **104**, using a plurality of homesite appliances. This plurality of homesite appliances can be load balanced, and set up to access the same storage array, so that a user can be directed to any of the homesite appliances and still be able to retrieve the user information. Furthermore, if a user is directed to the same homesite appliance **130** selected from the plurality, the prefetched and decrypted user data can be accessed without leaving the homesite appliance **130**, which reduces the time required to access the user profile.

The combination of the appliance design, and the encryption used to store user data, without storing the full decryption key, allows a homesite to offer secure storage to a user. In such a design, the homesite appliance **130** is unable to decrypt the data without user participation.

The homesite appliance of the present invention can provide hardware acceleration for suitable tasks to allow for a more efficient design, and as well provides enhanced security, which appeals to be homesite service providers and users. Homesite service providers can obtain assurances of security, and reduce the administration time to ensuring that appropriate security patches are applied. Users can be assured of the security of their data through the use of the storage side encryption, and by an assertion of security issued by a trusted authority.

In operation, a user provides user identity information to appliance **130** for storage. Appliance **130** stores a GUPI and user authentication information along with user identity information, so that the user associated with the GUPI can be later authenticated. At a later point in time the user can revisit the appliance and either provide more user identity

information, or can request that the appliance **130** release information. The request processor **136** receives this request for information release and preferably obtains user authorization for the release. The information is retrieved and extracted from the schema, preferably after retrieval from storage array **132**, using a storage array interface, and decryption by engine **134**. The decrypted data is preferably stored in a volatile memory which functions as a cache. The extracted data is then transmitted to the membersite requesting the data, preferably via redirection of browser **112**. In a presently preferred embodiment, either generic encryption assist hardware **135** or SSL encryption engine **138** is used to encrypt the data prior to it being sent to browser **112**. Appliance **130** also preferably includes an authoritative site interface to allow direct connection to an authoritative site to refresh or obtain an assertion, as well as an assertion checking module, which can be optionally included in request processor **136**, or executed as a separate software application by the same hardware processor, for checking assertions for validity and expiry.

The schema used in identity management system **100** is defined by the root, and shared with both membersites and homesites. The schema is used by homesites, such as HS1 **104** and HS2 **106** to store user data, and is used by membersites, such as MS **108**, to determine what data can be requested. A schema designed so that it can be expanded but never contracted, allows for a homesite to decide to not upgrade to a newer version of the schema, but at the same time still retain the ability to provide any stored user data to membersites without fear of incompatibility between the schema. Furthermore, when a schema is updated, a membersite that is already retrieving all the information that it wants will not need to consider the updated schema as relevant.

A homesite that adds new elements to the schema, does not have to prompt a user to add the data immediately. The user only needs to add the data when a membersite requests it. Thus, when MS **108** requests information about a user that is not stored in HS1 **104**, HS1 **104** can ask the user to provide this information. The information can then both be stored in the user profile and transmitted to MS **108**. This allows a just-in-time building of the user profile. The same technique can be used when the user is first registered into the network **100**, allowing a user to provide no information other than an email address to create a profile and obtain a GUPI.

- 33 -

Because the schema used to store the user profile is preferably an XML document, the user information in the schema is preferably related to a name or property field. The names of these properties, hereinafter referred to a schema names, can be mapped to user friendly names applied by the homesite, and preferably offered in different languages. When the schema is updated, a homesite can be provided with an easy to manage upgrade path to offer support for the schema. A mapping between schema names and user friendly names in different languages can be offered through the use of schema name mapping templates, such as those available through conventional Extensible Stylesheet Language Transformations (XSLT) techniques. An XSLT processor can be provided with an XSLT template that includes translations for the various schema names. A homesite can alter the XSLT template to customize it, or the XSLT template can be used in conjunction with another template to convert the standard schema into a document that can be viewed by the user. Because XSLT processors can read an XML document and write the data to another markup language format, a local language specific XSLT template can be used to translate the schema to a different language, and a homesite specific XSLT template can then be used to translate the local language schema to an HTML document for sending to users. This can either be done on the fly, or it can be done in advance for a number of pages to reduce the processing load on the homesite.

Membersites can request data in a rich format using XML Path Language (XPath) commands. One skilled in the art will appreciate that XPath commands allow a membersite to conditionally query the user's schema for information. For example, if the user's name, birthdate, and academic history are stored in the profile, MS **108** could request a user's name and birthdate. Alternately, MS **108** could request the user's name, birthdate, and date of university graduation only if the academic history indicated that the user had graduated from a university. MS **108** could also request the date of the most recent university graduation if there are more than one. Such structured queries allow for rich data requests which return rich XML responses. The homesite is also able to examine the XPath request and determine if it will return valid data for a user. This examination allows the homesite to prevent XPath requests that will return error messages or invalid data. XPath requests are used in the presently preferred embodiment because they allow for both simple and rich requests, and allow the homesite to handle only one type of request.

When XML fragments are transmitted from HS1 **104** to MS **108** through browser **112**, a simple mechanism is preferably selected as a standard transmission mechanism. In a presently preferable embodiment, the XML fragment is URL encoded and becomes a name value pair to be transmitted to MS **108** via a redirected post (as described above). The XML fragment as generated by HS1 **104** contains the information requested by MS **108**. To provide this information to MS **108** through browser **112**, the browser is send an HTML page containing the XML fragment, preferably in a hidden field. The HTML page contains a post link, which can be automatically executed in the case of the redirected post, designed to post the XML fragment to MS **108**. The use of the HTML post allows for large XML fragments, possibly including a series of assertions and other rich data, to be provided to MS **108** without the use of a back channel.

Requests sent by MS **108** to HS1 **104** are preferably URL encoded as name value pairs, where one value is and XML fragment. The rich response comes back to MS **108** as a name value pair, where a value is an entire XML fragment containing the requested data. A simple check on the response, performed by MS **108**, can determine if the response is validly formatted or is part of a distributed denial of service attack. If a content check is performed on the name value pairs and reveals that the response is invalid, further analysis of the XML code is unnecessary

Assertions and delegations of authority are described above. The present invention provides a novel mechanism to store assertions and chains of assertions. Conventional security markup languages, such as SAML, can be used to store assertions in the user profile. However, SAML stores the data being asserted inside the assertion, which is digitally signed for authentication purposes. Such as structure is illustrated below.

```
<profile>
     <assertion1>
          <data1 />
     </assertion1>
     <assertion2>
          <data2 />
     </assertion2>
</profile>
```

Thus, for a system like MS **108**, to read the data contained in the assertion, the assertion schema must be understood for the data can be read. This could mean requiring every membersite to support signed assertions so that they can access the data in the assertion. An alternate embodiment is also presented that does not require decoding an

assertion to access the asserted datablock. The schema structure illustrated below has data inline with the related assertions instead of being inside the assertion.

```
<profile>
      <data1>
            <datablock>
            <assertion of datablock />
            <digital signature />
      </data1>
</profile>
```

This allows the assertion to be separated from the datablock being asserted. As a result a business rule can be processed on the data without giving it the ability to verify the assertion. All assertions can be verified using a generic assertion checker that checks the assertion for validity, and checks the delegation chain. The decoupling of the data processing and assertion verification additionally allows for the context of a data element nested in other data elements to be maintained, whereas if it is embedded in an assertion the context of the data is lost. By keeping assertions inline with the data, instead of the data inside the assertions, a single method of requesting information, by its node, is available, thus simplifying the membersite-homesite interactions. Furthermore, data, along with the accompanying assertions, is received as a rich XML fragment, as opposed to being received as a series of assertions, each of which contains data.

The above-described embodiments of the present invention are intended to be examples only. Alterations, modifications and variations may be effected to the particular embodiments by those of skill in the art without departing from the scope of the invention, which is defined solely by the claims appended hereto.

What is claimed is:

1.   A homesite appliance for managing user identity information, the homesite appliance comprising:

     a request processor for receiving requests for the user identity information from
5   an external system, for authenticating a user associated with the user identity information and for obtaining authorization from the authenticated user to release the requested user identity information, for extracting the requested user identity information from a schema containing the user identity information, and for transmitting the extracted user identity information to the external system upon receipt of user
10   authorization.

2.   The homesite appliance of claim 1, further including a storage array interface for connecting the request processor to a storage array, the storage array for storing the schema containing the user identity information sent by the request processor, and for retrieving previously stored user identity information.

15   3.   The homesite appliance of claim 2, wherein the storage array is external to the homesite appliance.

4.   The homesite appliance of claim 2 further including a hardware encryption engine for encrypting user identity information sent to the storage array.

5.   The homesite appliance of claim 4, wherein the hardware encryption engine
20   includes means for decrypting user identity information retrieved from the storage array.

6.   The homesite appliance of claim 5, further including a cache memory for use by the request processor to cache decrypted user identity information.

7.   The homesite appliance of claim 6, further including a cache memory controller
25   for directing retrieval requests from the request processor to one of the cache memory and the external storage array.

8.   The homesite appliance of claim 1, wherein the request processor includes user authentication means for authenticating the identity of the user using at least one

- 37 -

of a request for a username and password combination, a request for an out of band authentication request for a shared secret, a request for biometric data and a request for a securely generated token.

9.   The homesite appliance of claim 1, wherein the request processor includes means to receive requests for the user identity information from the external system via the user associated with the user identity information.

10.   The homesite appliance of claim 1, wherein the request processor includes means to transmit the retrieved user identity information to the external system via the user associated with the user identity information.

11.   The homesite appliance of claim 1, wherein the request processor includes means to encrypt the retrieved user identity information prior to transmission to the user.

12.   The homesite appliance of claim 11, wherein the means to encrypt perform secure sockets layer encryption.

13.   The homesite appliance of claim 11, further including a secure sockets layer encryption engine for encrypting the retrieved user identity information prior to transmission.

14.   The homesite appliance of claim 4, wherein the hardware encryption engine includes means to encrypt the retrieved user identity information prior to transmission to the user.

15.   The homesite appliance of claim 14, wherein the means to encrypt the retrieved user identity information perform secure sockets layer encryption.

16.   The homesite appliance of claim 1, wherein the homesite appliance includes an authoritative site interface, for obtaining and refreshing third party assertions associated with the user identity information.
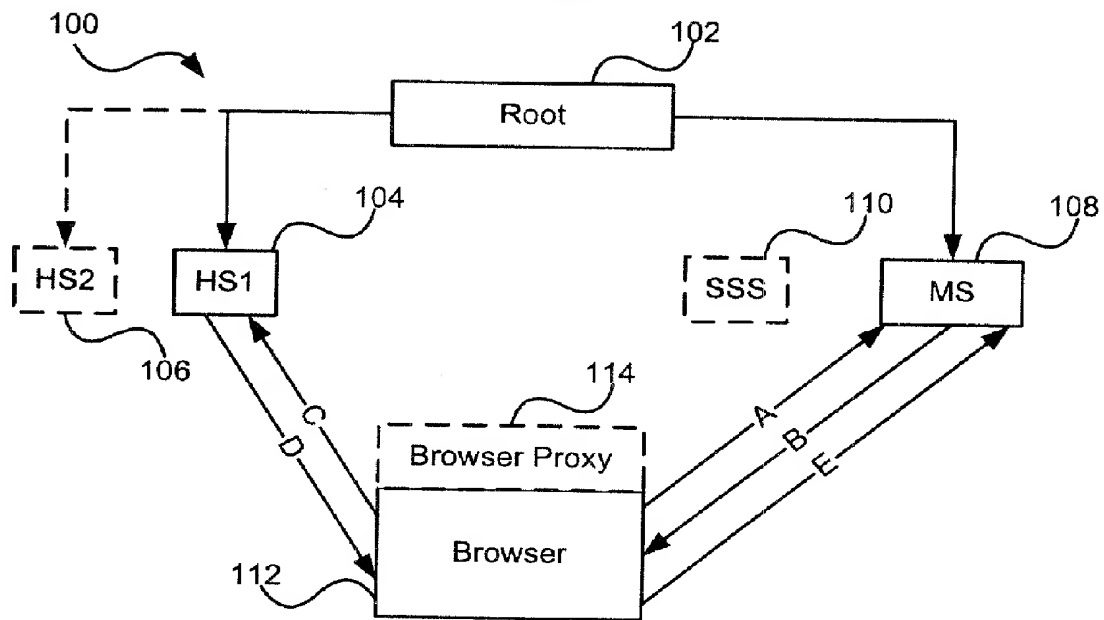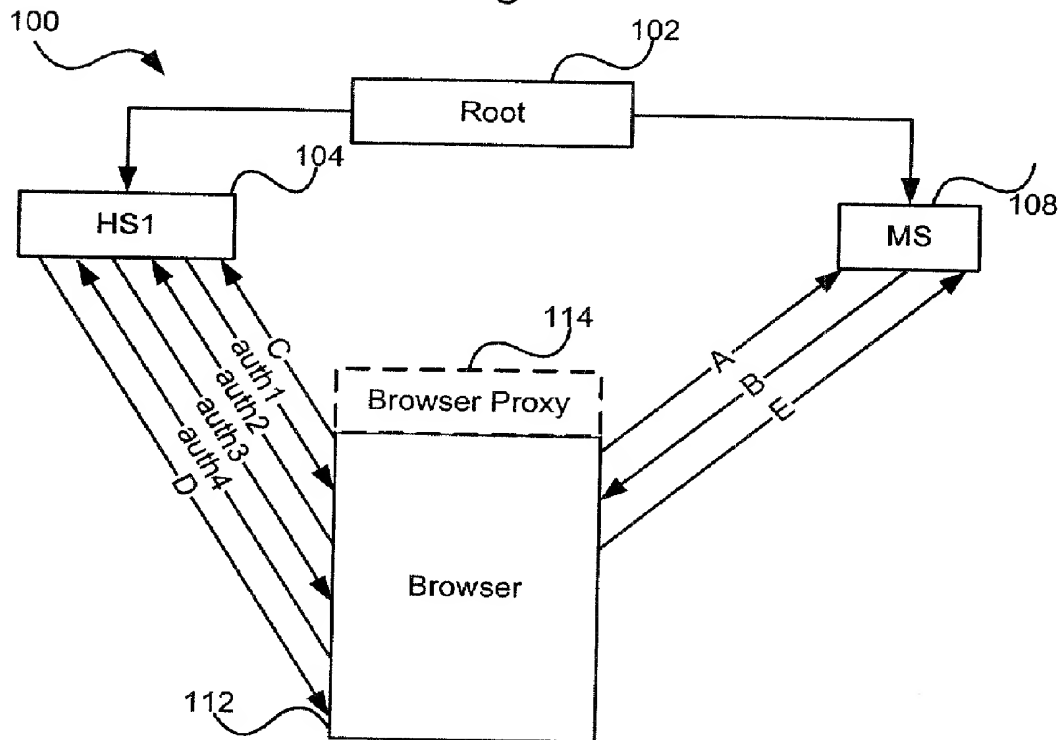
Figure 1

Figure 2

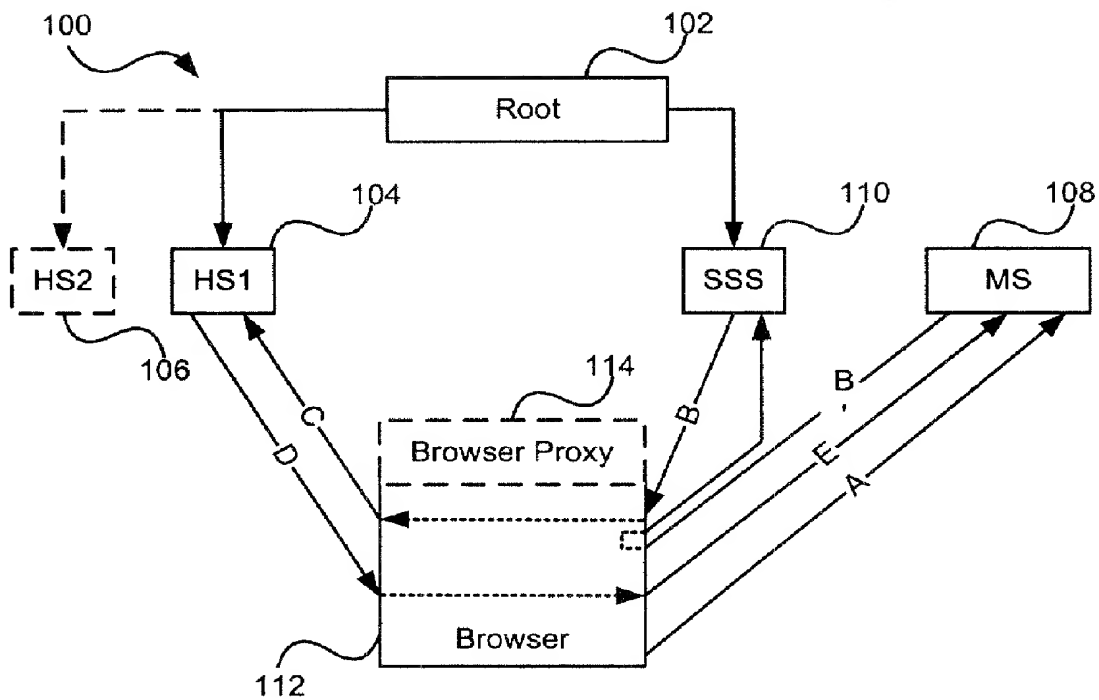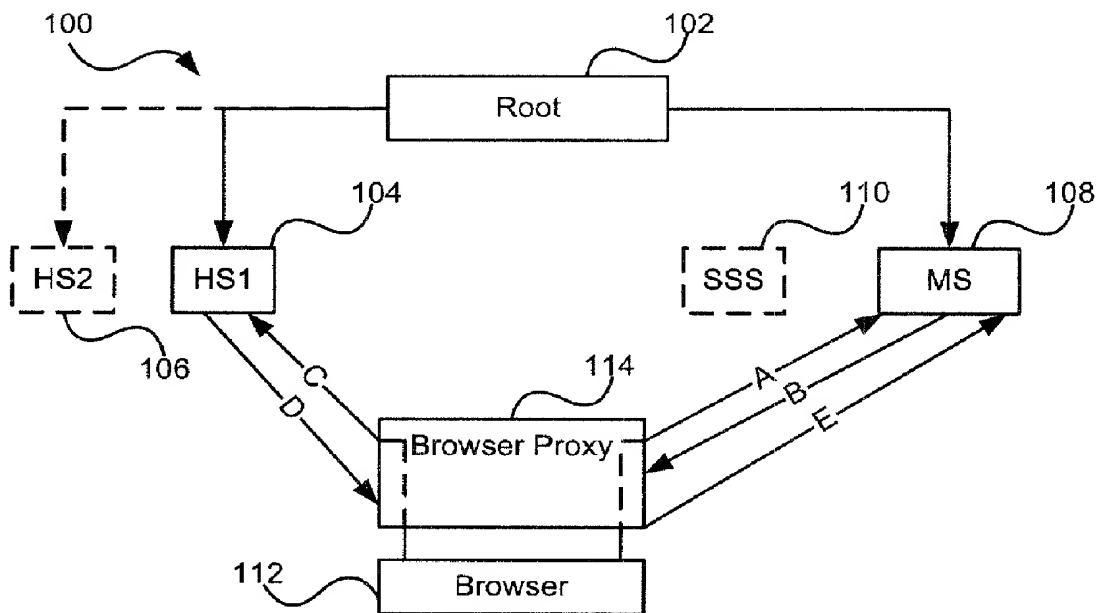Figure 3

Figure 4

3/9
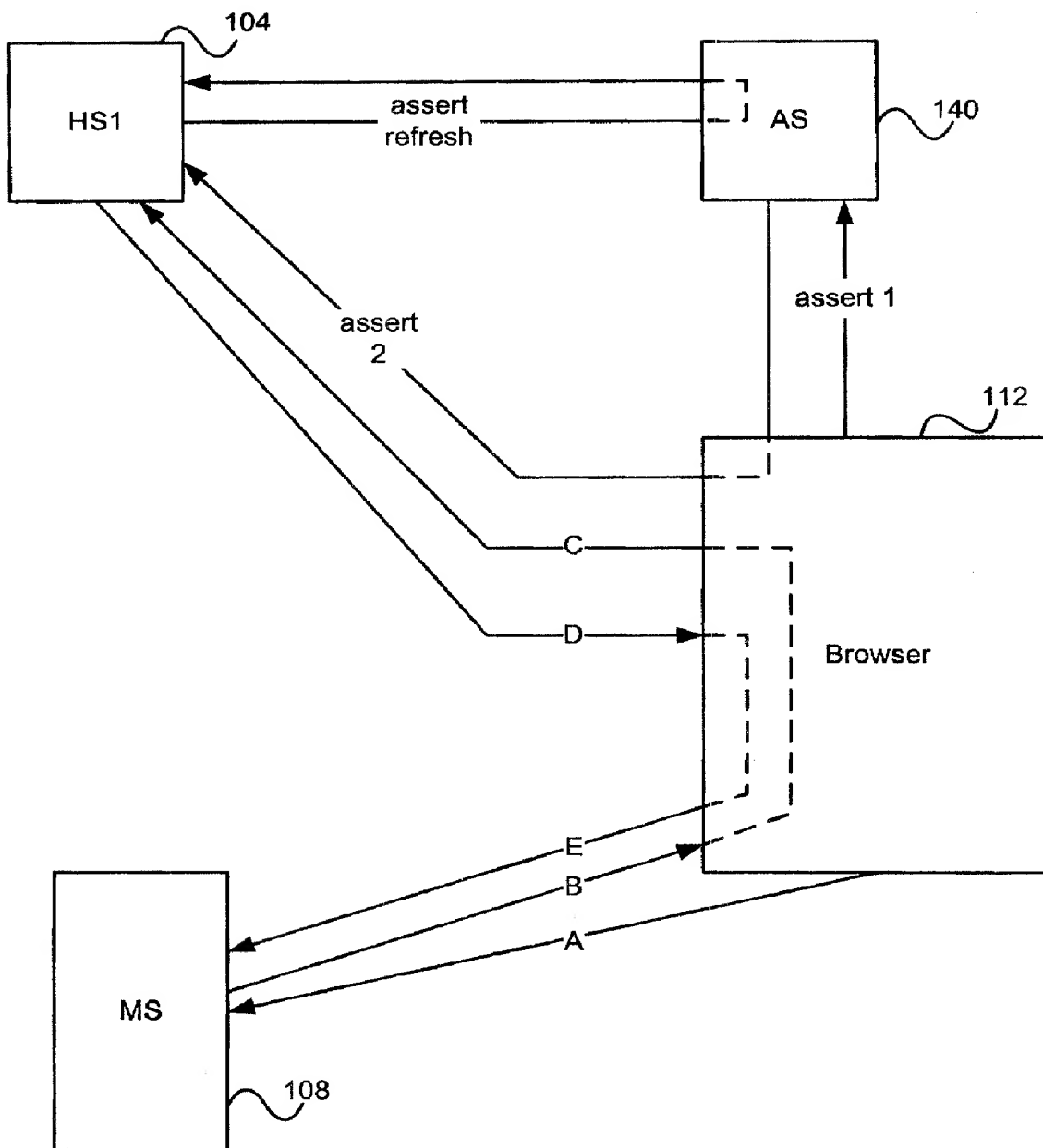
Figure 5

Figure 6

Figure 7

5/9

```
┌─────────────────────────────┐
│  Receive redirected user from │        150
│            MS               │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│    Redirect user to HS for   │        152
│       authentication         │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│     Receive redirected,      │        154
│  authenticated user from HS  │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│      Authenticate HS         │        156
│  authoritativeness for GUPI  │
└─────────────────────────────┘
              │
      158     ▼
◇─────────────────────────────◇
 ─Yes─  Is GUPI in account list?  ─No─
◇─────────────────────────────◇
   │                              │
   ▼                         162  ▼
┌──────────────────┐    Yes──◇──────────────◇
│ Match GUPI to     │         │  New user?   │
│ account           │         ◇──────────────◇
└──────────────────┘              │
        160            164    No  166
         │              │          │
         ▼              ▼          ▼
              ┌──────────────────┐  ┌──────────────┐
              │ Link GUPI to provided │ Create account │
              │ account information │  └──────────────┘
              └──────────────────┘
                      │                    │
                      ▼                    168
┌─────────────────────────────────────────────┐
│                  Login                       │
└─────────────────────────────────────────────┘
```

Figure 8

6/9

Figure 9

Receive privacy policy and request for user data — 178

Provide privacy policy to user — 180

Authenticate user — 182

Obtain user approval for user data release in accordance with the privacy policy — 184

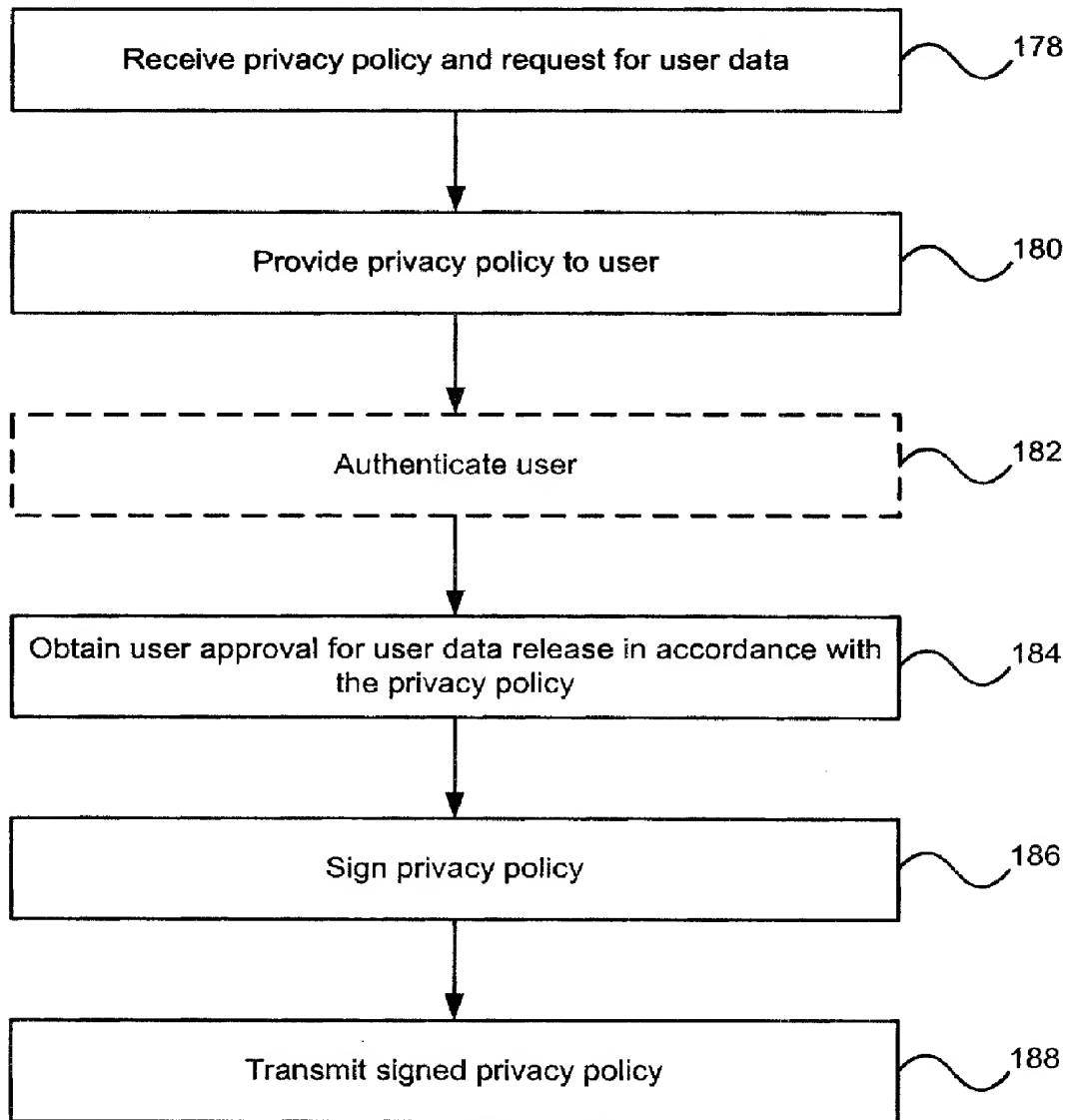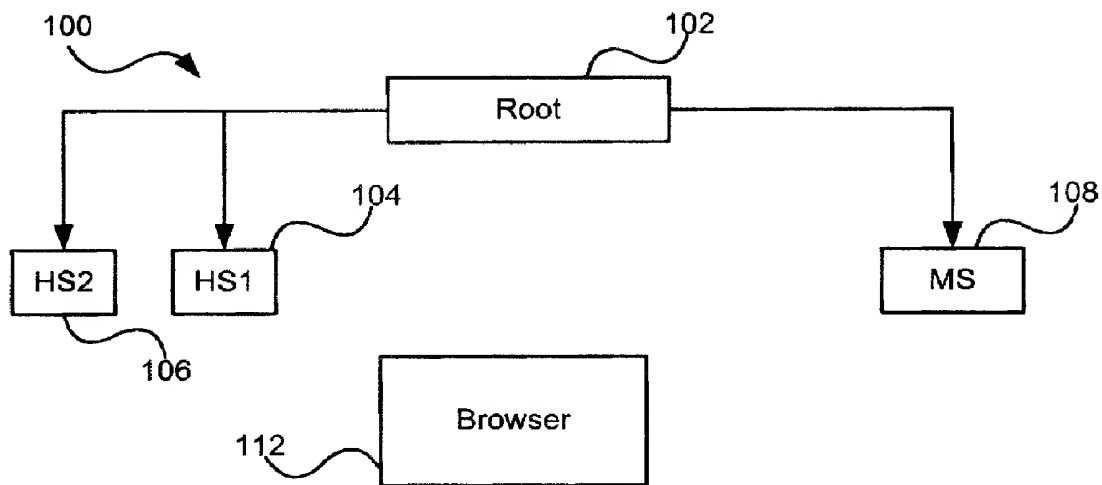Sign privacy policy — 186

Transmit signed privacy policy — 188

Figure 10

8/9

Figure 11

Figure 12

Figure 13